



INGENIERÍA DE SOFTWARE III

Eric Gustavo Coronel Castillo

gcoronelc.blogspot.com

ecoronel@gmail.com

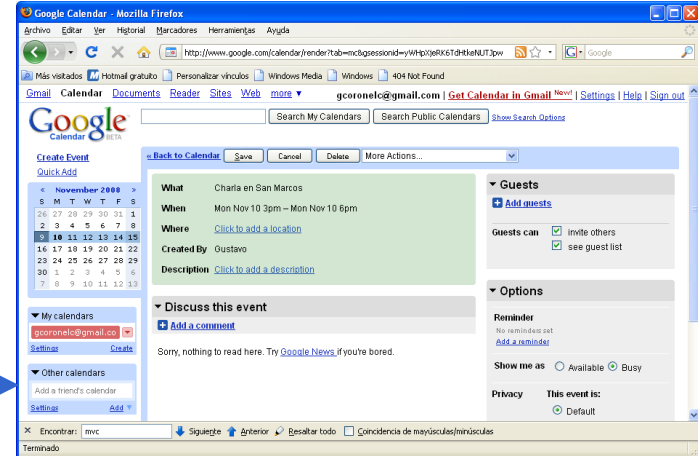
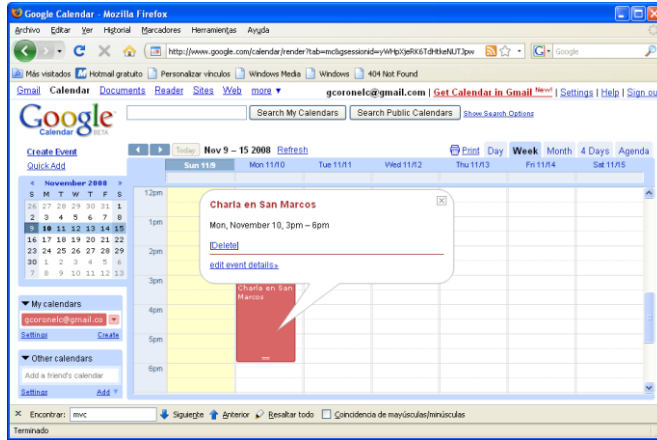


TEMA: AJAX

- Objetivo
- Definición
- Conversación Tradicional Cliente-Servidor
- Interacción AJAX Cliente-Servidor
- El Objeto XMLHttpRequest
- Implementación
- AJAX con JQuery



- Desarrollar aplicaciones Web que utilicen AJAX para mejorar su rendimiento y la experiencia de usuario (Usabilidad).

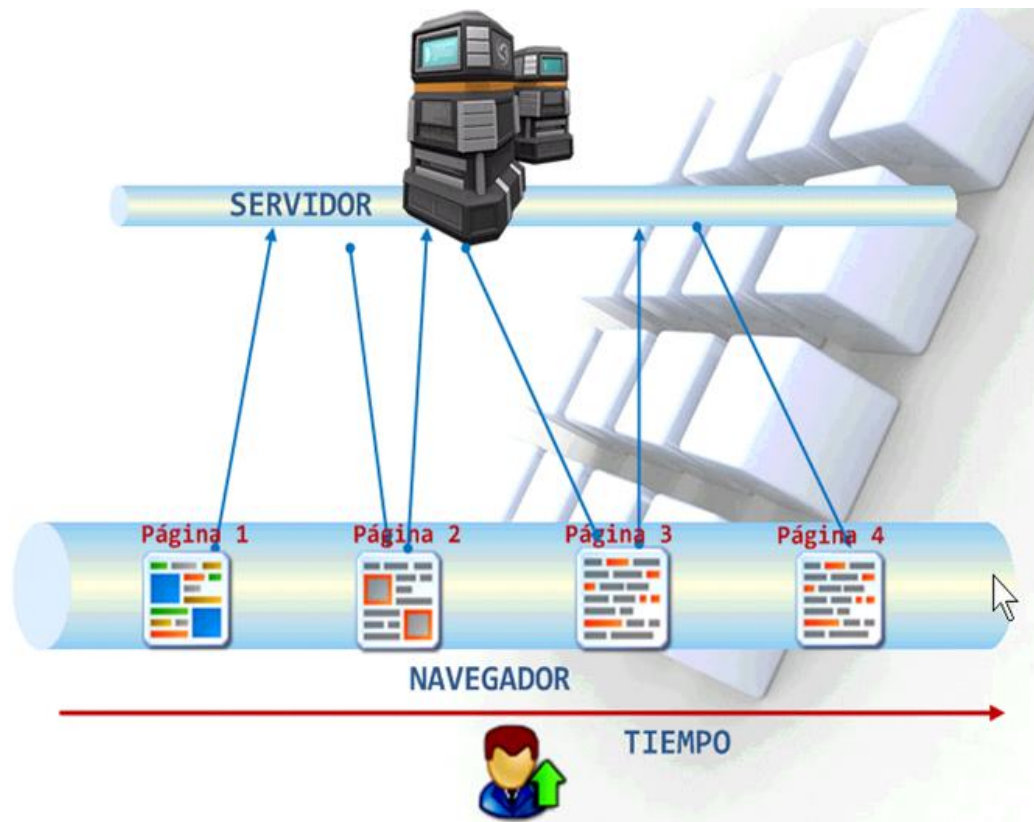


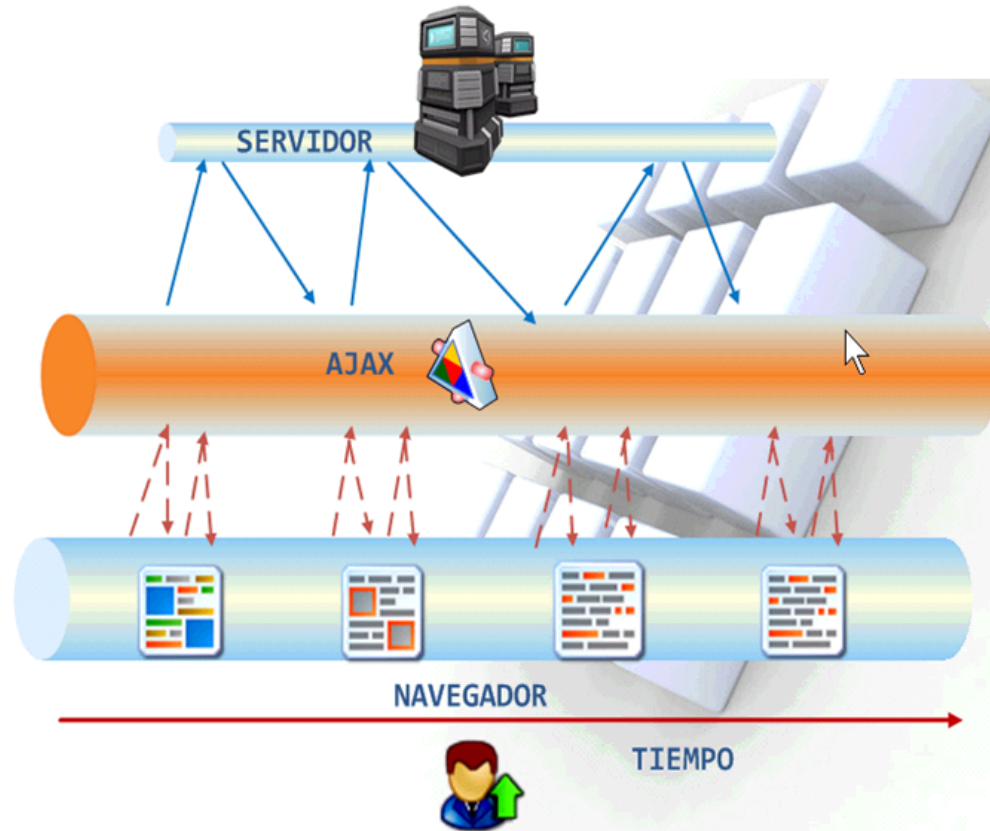


- AJAX, acrónimo de **A**synchronous **J**avaScript **A**nd **X**ML (**J**avaScript y **X**ML asíncronos, donde XML es un acrónimo de **eX**tensible **M**arkup **L**anguage).
- Es una técnica de desarrollo web para crear aplicaciones interactivas. Éstas se ejecutan en el cliente, es decir, en el navegador del usuario, y mantiene comunicación asíncrona con el servidor en segundo plano.
- De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la interactividad, velocidad y usabilidad en la misma.

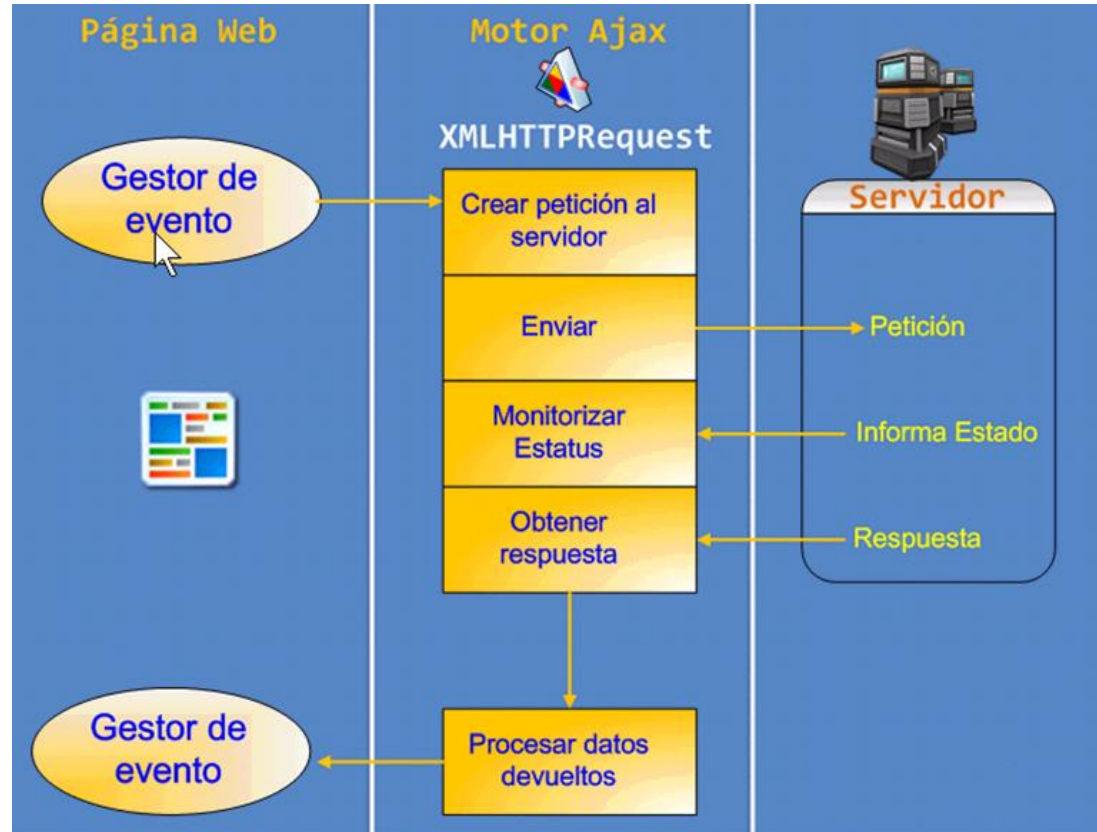


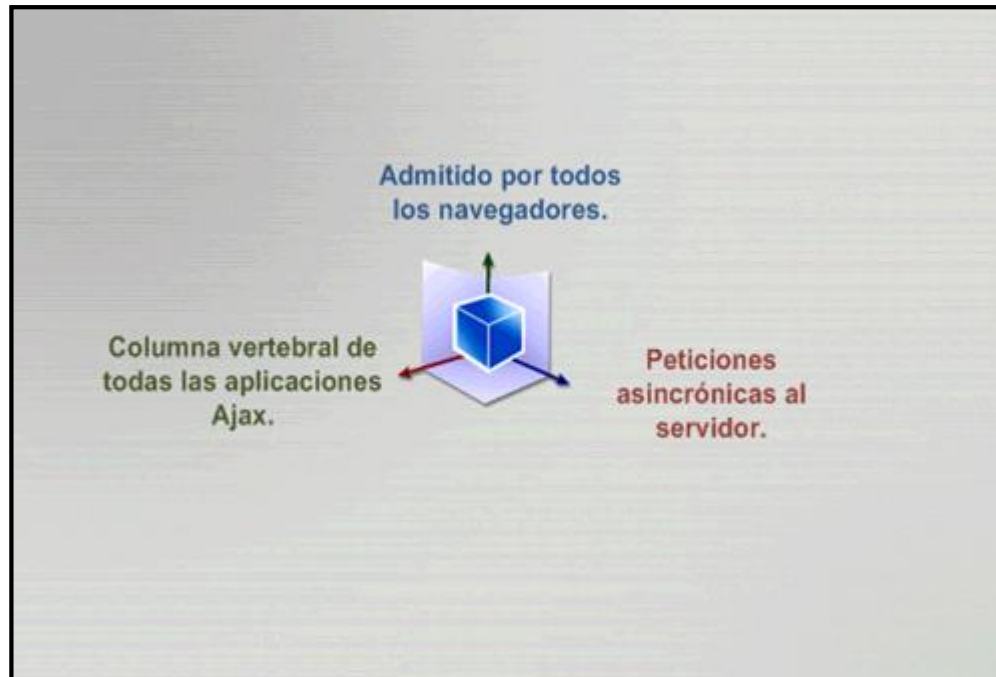
- **AJAX es la combinación de tres tecnologías ya existentes:**
 - **XHTML** y hojas de estilos en cascada (**CSS**) para el diseño que acompaña a la información.
 - Document Object Model (**DOM**) accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones de ECMAScript como JavaScript, para mostrar e interactuar dinámicamente con la información presentada.
 - El objeto **XMLHttpRequest** para intercambiar datos asincrónicamente con el servidor web.
 - **XML** es el formato usado comúnmente para la transferencia devuelta por el servidor, aunque cualquier formato puede funcionar, incluyendo HTML preformateado, texto plano, JSON y hasta EBML.

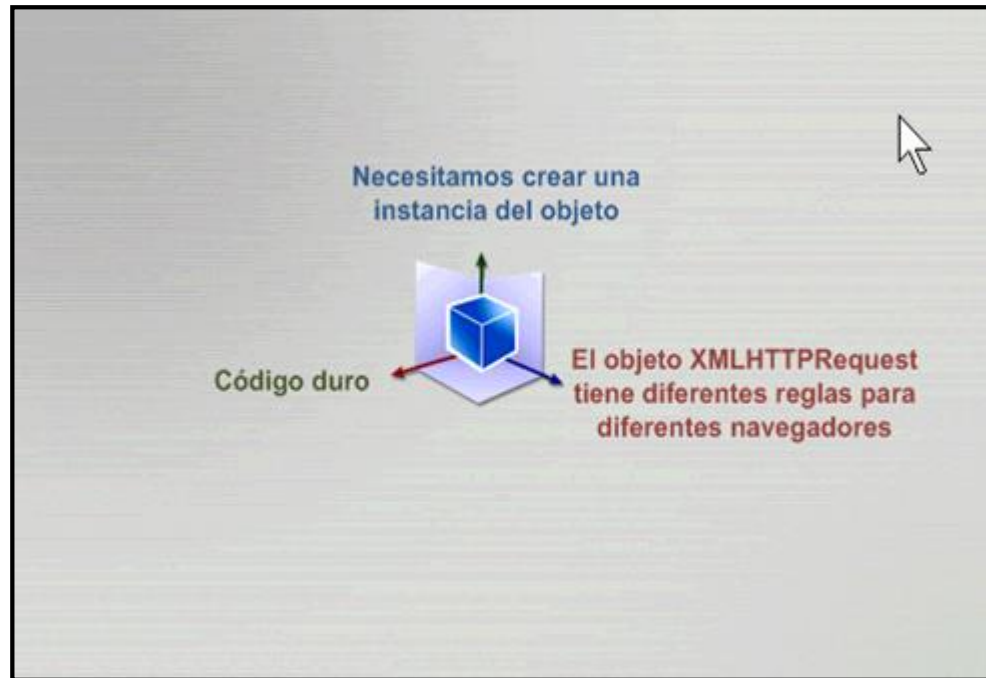




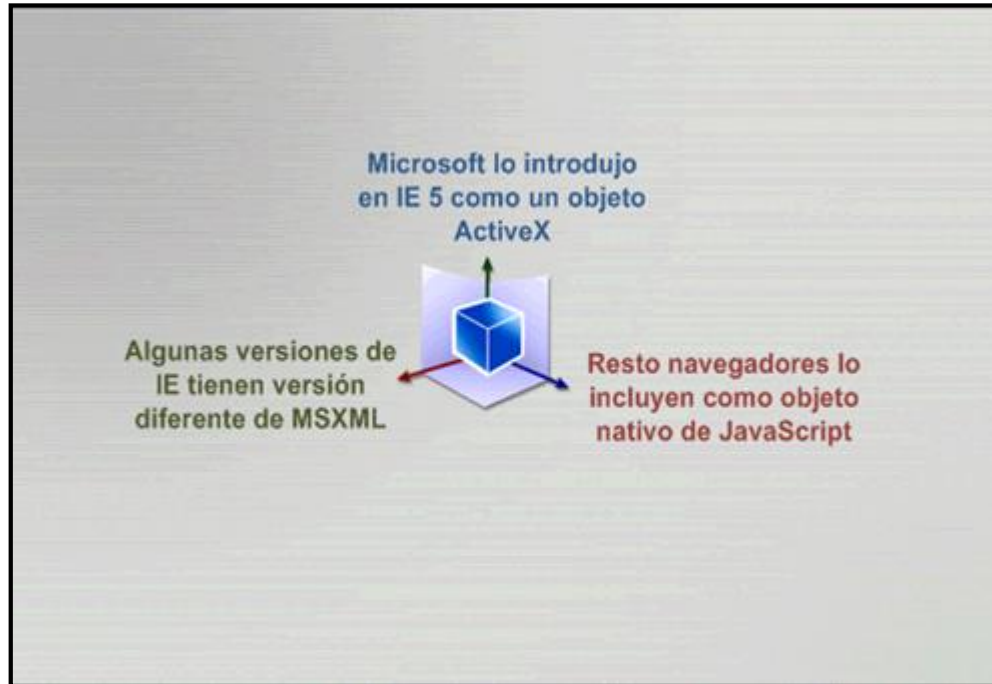




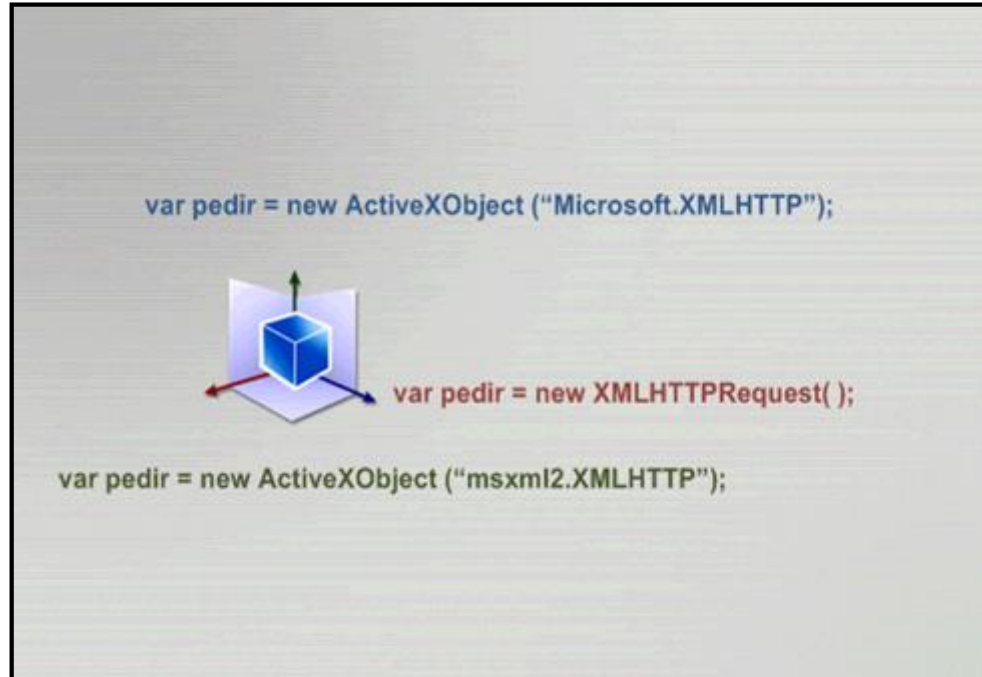




Diferentes reglas para diferentes navegadores.



Diferentes reglas para diferentes navegadores.





Diferentes reglas para diferentes navegadores.

```
function getXMLHttpRequest() {  
    var req;  
    try {  
        req = new XMLHttpRequest();  
    } catch(err1) {  
        try {  
            req = new ActiveXObject("Msxml2.XMLHTTP");  
        } catch (err2) {  
            try {  
                req = new ActiveXObject("Microsoft.XMLHTTP");  
            } catch (err3) {  
                req = false;  
            }  
        }  
    }  
    return req;  
}
```

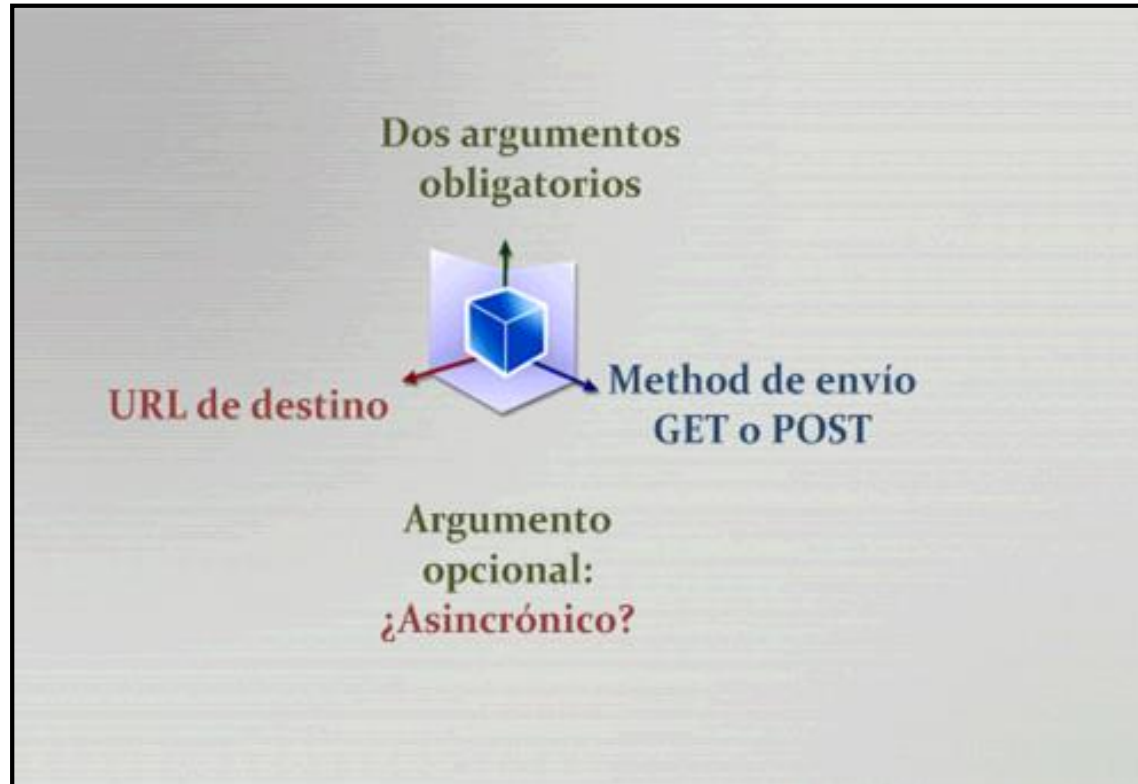
Propiedades del objeto XMLHttpRequest

Propiedad	Descripción
onreadystatechange	Determina el gestor de evento que será llamado cuando la propiedad readyState del objeto cambie.
readyState	Número entero que indica el estado de la petición: 0: No iniciada 1: Cargando 2: Cargado 3: Interactivo 4: Completado
responseText	Datos devueltos por el servidor como una cadena de texto.
responseXML	Datos devueltos por el servidor como un objeto XMLDocument que se puede recorrer usando las funciones de JavaScript DOM.
status	Código de estado HTTP devuelto por el servidor.
statusText	Mensaje de texto enviado por el servidor junto al código (status), para el caso de código 200 contendrá "OK".

Métodos del objeto XMLHttpRequest

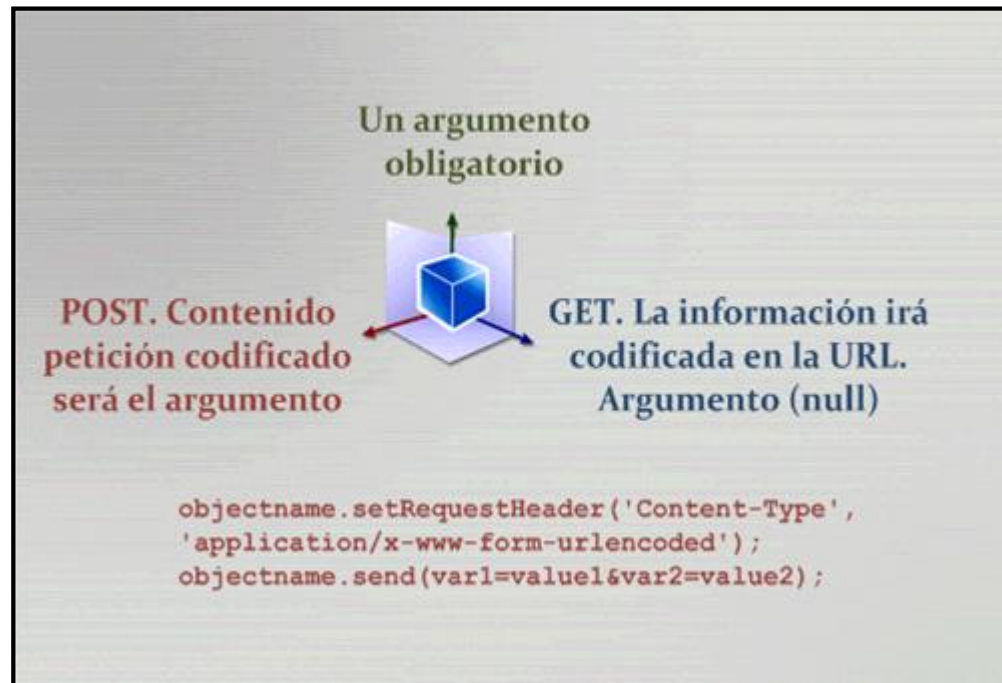
Propiedad	Descripción
abort()	Detiene la petición actual.
getAllResponseHeaders()	Devuelve todas las cabeceras como una cadena.
getResponseHeader(etiqueta)	Devuelve el valor de la etiqueta en las cabeceras de la respuesta.
open(método,URL,asíncrona)	Especifica un método HTTP (GET o POST), la URL objetivo, y si la petición debe ser manejada asincrónicamente (true o false).
send(contenido)	Envía la petición, opcionalmente con datos POST.
setRequestHeader(etiqueta,valor)	Establece el valor de una etiqueta de las cabeceras de petición.

- Método: open(método, URL, asíncrona)





Método: send(contenido)





• Ejemplo 01

En este ejemplo se está enviando los datos utilizando el método GET y se está recibiendo un texto.

formulario.html

Suma de Dos Números

Número 1	<input type="text"/>
Número 2	<input type="text"/>
Suma	<input type="text"/>
<input type="button" value="Calcular"/> <input type="button" value="Limpiar"/>	

www.perudev.net

procesar.jsp

```
<%  
// Datos  
int n1 = Integer.parseInt( request.getParameter("n1") );  
int n2 = Integer.parseInt( request.getParameter("n2") );  
// Proceso  
int suma = n1 + n2;  
// Reporte  
out.print(suma);  
%>
```



• Ejemplo 02

En este ejemplo se está enviando los datos utilizando el método POST, se esta recibiendo un documento XML y la función de procesamiento se esta construyendo en línea.

procesar.jsp

```
<%@page contentType="text/xml" %>
<%
// Datos
int n1 = Integer.parseInt( request.getParameter("n1") );
int n2 = Integer.parseInt( request.getParameter("n2") );
// Proceso
int suma = n1 + n2;
// Reporte
out.print( "<rpta><suma>" );
out.print( suma );
out.print( "</suma></rpta>" );
%>
```

formulario.html

Suma de Dos Números

Número 1

Número 2

Suma

Calcular

Limpiar

www.perudev.net



• Ejemplo 03

En este ejemplo se está enviando los datos utilizando el método POST, luego se retorna el resultado en formato **JSON** (Arreglo) y la función de procesamiento está utilizando la función **eval()** para convertir el resultado en un arreglo JavaScript.

procesar.jsp

```
<%
// Datos
int n1 = Integer.parseInt( request.getParameter("n1") );
int n2 = Integer.parseInt( request.getParameter("n2") );
// Proceso
int suma = n1 + n2;
// Reporte
out.print( "[" + suma + "]" );
%>
```

formulario.html

Suma de Dos Números

Número 1

Número 2

Suma

www.perudev.net



• Ejemplo 04

En este ejemplo se está enviando los datos utilizando el método POST, luego se retorna el resultado en formato **JSON** (Object) y la función de procesamiento está utilizando la función **eval()** para convertir el resultado en un objeto JavaScript.

procesar.jsp

```
<%
// Datos
int n1 = Integer.parseInt( request.getParameter("n1") );
int n2 = Integer.parseInt( request.getParameter("n2") );
// Proceso
int suma = n1 + n2;
// Reporte
out.print( "{ valor:" + suma + "}" );
%>
```

formulario.html

Suma de Dos Números

Número 1

Número 2

Suma

Calcular

Limpiar

www.perudev.net



• Ejemplo 05

En este ejemplo se está enviando los datos utilizando el método POST y retorna una porción de código HTML que es visualizado en un **div** de nombre **divresultado**.

formulario.html

Suma de Dos Números

Número 1

17

Número 2

23

Calcular

Limpiar

Resultado

Número 1	17
Número 2	23
Suma	40

www.perudev.net

procesar.jsp

```
<%
// Datos
int n1 = Integer.parseInt(request.getParameter("n1"));
int n2 = Integer.parseInt(request.getParameter("n2"));
// Proceso
int suma = n1 + n2;
%>
<h2>Resultado</h2>
<table width="190">
  <tr>
    <td width="82">Número 1</td>
    <td width="96"><%=n1%></td>
  </tr>
  <tr>
    <td>Número 2</td>
    <td><%=n2%></td>
  </tr>
  <tr>
    <td>Suma</td>
    <td><%=suma%></td>
  </tr>
</table>
```




- La forma más sencilla de enviar una petición HTTP de forma asíncrona y mostrar el resultado en la página actual es utilizar la función `load()`. Esta se ejecuta sobre el elemento al que se va a añadir la respuesta, y se le pasa como argumento una cadena con el archivo a cargar:

```
$("#divResultado").load("productos.jsp");
```



- También se le puede enviar parámetros:

```
$("#divResultado").load("productos.jsp","cate=15");
```

- También se pueden utilizar los métodos `get()` y `post()` del objeto `jQuery`, en cuyo caso devolverá un resultado con los que tendrás que trabajar para generar la respuesta y mostrarla en el documento actual:



```
$.get("LogonController", {nombre: "gcoronelc",  
pass:"secreto"}, function(data, textStatus,  
XMLHttpRequest){  
    $("#mensaje").html("Han devuelto: " + data);  
});
```

```
$.post("LogonController", {nombre: "gcoronelc",  
pass:"secreto"}, function(data, textStatus,  
XMLHttpRequest){  
    $("#mensaje").html("Han devuelto: " + data);  
});
```



- En estos momentos AJAX está revolucionando el modo en el que se programan las webs y el modo en el que se presentan. Gracias a AJAX se consigue una interfaz muy parecida a la de una aplicación de escritorio.
- La evolución de AJAX ha sido un hecho que ha marcado la tendencia en los lenguajes de programación en estos últimos años.
- Las aplicaciones de AJAX utilizan características bien documentadas presentes en todos los navegadores importantes en la mayoría de las plataformas existentes. Aunque esta situación podría cambiar en el futuro, en este momento, el uso de AJAX es efectivo entre plataformas.
- Para hacer mas productivo el desarrollo con AJAX se recomienda el uso de Frameworks; como por ejemplo: Prototype, Scriptaculous, Xajax, jQuery, etc.